EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L2	183	instrument$5 near5 ((type near3 check$3) or (class near3 (check$3 or validation or verification or valid)) or (incompatible near5 type))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/14 07:51
L3	169	instrument$5 near5 (type near3 check$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/14 07:53
L4	14	l2 not l3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/14 07:53
S1	9	type adj (test$3 or check$3) same static$3 and (optimal$2 or optimiz$5 or optimis$5) and inlin$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:24
S2	20	("7080354" "6948156" "6978448" "5999732" "5579518" "5345384" "6892212" "5995754" "7120572" "6658657" "6658657" "6079032" "6072951" "6317872" "5701489" "6971091" "6170083" "5361351" "6760907").pn. and (determin$3 or calculat$4) and ("number of" or threshold or benchmark or "less than" or more or less$2 or exceed$3 or minimiz$5 or count)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:31
S3	16	("7080354" "6948156" "6978448" "5999732" "5579518" "5345384" "6892212" "5995754" "7120572" "6658657" "6658657" "6079032" "6072951" "6317872" "5701489" "6971091" "6170083" "5361351" "6760907").pn. and (determin$3 or calculat$4 or profil$3) same ("number of" or threshold or benchmark or "less than" or more or less$2 or exceed$3 or minimiz$5 or count)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:32

EAST Search History

S4	16	("7080354" "6948156" "6978448" "5999732" "5579518" "5345384" "6892212" "5995754" "7120572" "6658657" "6658657" "6079032" "6072951" "6317872" "5701489" "6971091" "6170083" "5361351" "6760907").pn. and (determin$3 or calculat$4 or profil$3) same ("number of" or threshold or benchmark or "less than" or more or less$2 or exceed$3 or minimiz$5 or count or metric or statistic$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:40
S5	3847	(determin$3 or calculat$4 or profil$3 or sampl$3 or instrument$5) same (inlin$3 or "in-line" or "in-lining") same ("number of" or threshold or benchmark or "less than" or more or less$2 or exceed$3 or minimiz$5 or count or metric or statistic$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:42
S6	68	(717/14?.ccls. or 717/15?.ccls.) and (determin$3 or calculat$4 or profil$3 or sampl$3 or instrument$5) same (inlin$3 or "in-line" or "in-lining") same ("number of" or threshold or benchmark or "less than" or more or less$2 or exceed$3 or minimiz$5 or count or metric or statistic$4)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/10/25 13:43
S7	64	S6 not S4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 16:56
S8	122	class near2 ((type adj check$3) or validation or verification or valid) and (inlin$3 or "in-line" or "code expansion")	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 16:58
S9	20	class near2 ((type adj check$3) or validation or verification or valid) and (inlin$3 or "in-line" or "code expansion") and (type near3 (frequency or frequencies or count or "number of")) and (cost or optimal$2 or optimiz$5 or optimis$5 or threshold or maximum)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:17

EAST Search History

S10	1	class near2 ((type adj check$3) or validation or verification or valid) and (inlin$3 or "in-line" or "code expansion") and (type near3 (frequency or frequencies or count or "number of")) and (cost or optimal$2 or optimiz$5 or optimis$5 or threshold or maximum) and (call$3 or invok$3 or invocation or branch$3 or jump$3) same class near2 ((type adj check$3) or validation or verification or valid)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:01
S11	1	profil$3 same class near2 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) near5 (inlin$3 or "in-line" or "code expansion" or (class near3 check$3) or validation or verification or valid) and ((class or type) near3 (frequency or frequencies or count or "number of")) and (cost or optimal$2 or optimiz$5 or optimis$5 or threshold or maximum or best)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:20
S12	1	profil$3 same class near5 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) near5 (inlin$3 or "in-line" or "code expansion" or (class near3 check$3) or validation or verification or valid) and ((class or type) near3 (frequency or frequencies or count or "number of")) and (cost or optimal$2 or optimiz$5 or optimis$5 or threshold or maximum or best)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:21
S13	16	(link$3 or linktime or runtime or profil$3 or instrument$5) same class near5 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) same (((replac$3 near3 call$3) near3 "with function") or inlin$3 or "in-line" or "code expansion" or (class near3 (check$3 or validation or verification or valid))) and ((class or type) near3 (frequency or frequencies or count or "number of")) and (cost or optimal$2 or optimiz$5 or optimis$5 or threshold or maximum or best)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:27

EAST Search History

S14	0	717/130.ccls. and (link$3 or linktime or runtime or profil$3 or instrument$5) same class near5 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) same (((replac$3 near3 call$3) near3 "with function") or inlin$3 or "in-line" or "code expansion" or (class near3 (check$3 or validation or verification or valid)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:28
S15	1	717/131.ccls. and (link$3 or linktime or runtime or profil$3 or instrument$5) same class near5 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) same (((replac$3 near3 call$3) near3 "with function") or inlin$3 or "in-line" or "code expansion" or (class near3 (check$3 or validation or verification or valid)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:29
S16	2	717/141.ccls. and (link$3 or linktime or runtime or profil$3 or instrument$5) same class near5 ((type adj check$3) or validation or verification or valid) and (add$3 or generat$3 or creat$3 or patch$3) same (((replac$3 near3 call$3) near3 "with function") or inlin$3 or "in-line" or "code expansion" or (class near3 (check$3 or validation or verification or valid)))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:33
S17	23111	(add$3 or generat$3 or creat$3 or patch$3 or instrument$5) same (((replac$3 near3 call$3) near3 "with function") or inlin$3 or "in-line" or "code expansion" or (class near3 (check$3 or validation or verification or valid)) or (incompatible near5 type))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:36
S18	374	(add$3 or generat$3 or creat$3 or patch$3 or instrument$5) same (inlin$3 or "in-line" or "code expansion") and ((class near3 (check$3 or validation or verification or valid)) or (incompatible near5 type) or (type near2 check$3))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 17:38

EAST Search History

S19	134	(add$3 or generat$3 or creat$3 or patch$3 or instrument$5) near5 (inlin$3 or "in-line" or "code expansion") and ((class near3 (check$3 or validation or verification or valid)) or (incompatible near5 type) or (type near2 check$3))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2007/03/13 18:11
S20	31	(add$3 or generat$3 or creat$3 or patch$3 or instrument$5) near5 (link$3 near2 (code or instruction)) and (inlin$3 or "in-line" or "code expansion") and ((class near3 (check$3 or validation or verification or valid)) or (incompatible near5 type) or (type near2 check$3))	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/13 18:19
S21	169	instrument$5 near5 (type near3 check$3)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	ON	2007/03/14 07:52

Google Scholar BETA

instrumenting link code inline Search

Advanced Schol
Scholar Preferer
Scholar Help

Scholar All articles Recent articles Results **1 - 10** of about **2,790** for **instrumenting link**

All Results

A Srivastava
A Eustace
B Ainsworth
W Haskell
M Whitt

[BOOK] ATOM: a system for building customized program analysis tools - group of 2 »
A Srivastava, A Eustace - 1994 - ACM Press New York, NY, USA
... ATOM, using OM **link**-time technology, organizes the fi ... tools such as Tango Lite, which
instrument assem- bly ... **Instrumenting** library routines is inconvenient as ...
Cited by 686 - Related Articles - Web Search - Library Search - BL Direct

A pooled analysis of magnetic fields, wire codes, and childhood leukemia - group of 5 »
S Greenland, AR Sheppard, WT Kaune, C Poole, MA … - Epidemiology, 2000 - epidem.com
... at a time and make instrumental (not **instrument**) variable corrections. ... A pooled analysis
of magnetic fields, wire **codes**, and childhood ... [Context **Link**]. 6. Ahlbom ...
Cited by 143 - Related Articles - Web Search - BL Direct

Link-time and run-time error detection, and program instrumentation - group of 2 »
DR Chase, SC Kendall, MP Mitchell - US Patent 6,149,318, 2000 - Google Patents
... The **link**-time error checking diag -noses violations of the ... To add **instrumentation**
to aC or C++ program, pre ... syntax tree before a back-end generates **code** for the ...
Cited by 10 - Related Articles - Web Search

Intercepting and **Instrumenting** COM Applications - group of 11 »
GC Hunt, ML Scott - usenix.org

... At **link** time, the linker embeds in the ... the cost of redirection, but not any additional
instrumentation. ... than DLL redirection or application **code** modification. ...
Cited by 18 - Related Articles - Web Search

Mobile **code** security by Java bytecode instrumentation - group of 16 »
A Chander, JC Mitchell, I Shin - 2001 DARPA Information Survivability Conference & Exposition ..., 2001 - doi.ieeecs.org
... The bytecode **instrumentation** technique itself is presented in Section ... proxies are
aug- mented to **instrument** the component ... a web page, like a graphic or a **link**. ...
Cited by 25 - Related Articles - Web Search

Minimum Data Set for Home Care: A Valid Instrument to Assess Frail Older People Living in the ...
M Instrument, S Analyses - Medical Care, 2000 - lww-medicalcare.com
... 300 items include many triggers that **link** the MDS ... a very simple and efficient assessment
instrument for elderly ... patients had a severe impairment (**code** 4 = total ...
Cited by 48 - Related Articles - Web Search - BL Direct

BIT: A Tool for **Instrumenting** Java Bytecodes - group of 8 »
HB Lee, BG Zorn - USENIX Symposium on Internet Technologies and Systems, 1997 - usenix.org
... Java world by allowing a user to **instrument** a JVM ... BIT: Bytecode **Instrumenting**= Tool ...
"A Practical System for Intermodule **Code** Optimization at **Link**-Time." Journal ...
Cited by 79 - Related Articles - Web Search

Teaching the **Code** Book: Preparation for Data Entry - group of 3 »
ZR Wolf - Nurse Educator, 2003 - nurseeducatoronline.com
... from **code** book exercises to appreciate the **link** between instruments and data analysis.
Each semester that I teach the graduate course, I change the **instrument** ...

Web Search - BL Direct

Dynamic Binary Instrumentation for Intel® Itanium™ Processor Family
V Ramasamy, R Hundt - EPICI Workshop, Micro - h21007.www2.hp.com
... Run & **Instrument** Run & **Instrument** ... Binary reader – The binary, which is in the Executable
and **Link** Format (ELF) on HP-UX ... Out-of-line **instrumentation** with the ...
Cited by 1 - Related Articles - View as HTML - Web Search

DIOTA: Dynamic instrumentation, optimization and transformation of applications - group of 4 »
J Maebe, M Ronsse, K De Bosschere - Compendium of Workshops and Tutorials held in conjunction ..., 2002 - escher.elis.rug.ac.be
... **Instrument** until ... such an instruction has been encountered and processed, DIOTA stops
instrumenting and jumps ... The used offset is calculated at **link** time so that ...
Cited by 32 - Related Articles - View as HTML - Web Search

Goooooooooogle ▶
Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

instrumenting link code inline Search

Google Home - About Google - About Google Scholar

PORTAL

USPTO

Subscribe (Full Service) Register (Limited Service, Free)

Search: [✓ The ACM Digital Library] The Guide

+determining +inlin*

Feedback Report a problem Satisfaction s

Published since January 1980 and Published before September 2003

Terms used **determining** **inlin**

Found **1,988**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM

Results 1 - 20 of 200

Best 200 shown

Result page: **1** 2 3 4 5 6 7 8 9 10 next

Relevance scal

1 Inline function expansion for compiling C programs

P. P. Chang, W.-W. Hwu

June 1989 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1989 Conf on Programming language design and implementation PLDI '89**, Volum Issue 7

Publisher: ACM Press

Full text available: pdf(1.14 MB) Additional Information: full citation, abstract, referenc citings, index terms

Inline function expansion replaces a function call with the function body. With autom inline function expansion, programs can be constructed with many small functions to complexity and then rely on the compilation to eliminate most of the function calls. Therefore, inline expansion serves a tool for satisfying two conflicting goals: minizin complexity of the program development and minimizing the function call overhead o program execution. A simple inline expansion procedur ...

2 Run-time evaluation of opportunities for object inlining in Java

Ondrej Lhoták, Laurie Hendren

November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java G JGI '02**

Publisher: ACM Press

Full text available: pdf(188.19 KB) Additional Information: full citation, abstract, referenc citings, index terms

Object-oriented languages, such as Java, encourage the use of many small objects link together by field references, instead of a few monolithic structures. While this practic beneficial from a program design perspective, it can slow down program execution by incurring many pointer indirections. One solution to this problem is object inlining: w the compiler can safely do so, it fuses small objects together, thus removing the reads/ to the removed field, saving the memory needed to ...

Keywords: Java, compilers, object inlining, optimization

3 Flow-directed inlining
Suresh Jagannathan, Andrew Wright
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 conf on Programming language design and implementation PLDI '96**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, referenc citings, index terms

A *flow-directed inlining* strategy uses information derived from control-flow analysis specialize and inline procedures for functional and object-oriented languages. Since it control-flow analysis to identify candidate call sites, flow-directed inlining can inline procedures whose relationships to their call sites are not apparent. For instance, proce defined in other modules, passed as arguments, returned as values, or extracted from c structures can all be inlined. Flow-d ...

4 Function inlining under code size constraints for embedded processors
Rainer Leupers, Peter Marwedel
November 1999 **Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design ICCAD '99**
Publisher: IEEE Press
Full text available: pdf(184.10 KB) Additional Information: full citation, abstract, referenc citings, index terms

Function inlining is a compiler optimization that generally increases performance at tl expense of larger code size. However, current inlining techniques do not meet the spe demands in the design of embedded systems, since they are based on simple heuristic: they generate code of unpredictable size. This paper presents a novel approach to func inlining in C compilers for embedded processors, which aims a maximum program sp under a global limit on code size. The co ...

5 Dynamic Adaptive compilation: Adaptive online context-sensitive inlining
Kim Hazelwood, David Grove
March 2003 **Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization CGO '03**
Publisher: IEEE Computer Society
Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, referenc citings, index terms

As current trends in software development move toward more complex object-oriente programming, inlining has become a vital optimization that provides substantial performance improvements to C++ and Java programs. Yet, the aggressiveness of the inlining algorithm must be carefully monitored to effectively balance performance an size. The state-of-the-art is to use profile information (associated with call edges) to g inlining decisions. In the presence of virtual method calls, profile ...

6 An evaluation of automatic object inline allocation techniques
Julian Dolby, Andrew A. Chien
October 1998 **ACM SIGPLAN Notices , Proceedings of the 13th ACM SIGPLAN conference on Object-oriented programming, systems, languages, anc applications OOPSLA '98**, Volume 33 Issue 10
Publisher: ACM Press
Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, referenc citings, index terms

Object-oriented languages such as Java and Smalltalk provide a uniform object refere: model, allowing objects to be conveniently shared. If implemented directly, these uni reference models can suffer in efficiency due to additional memory dereferences and memory management operations. Automatic *inline allocation* of child objects within objects can reduce overheads of heap-allocated pointer-referenced objects.We present compiler analyses to identify inlinable fields by t ...

7 The effectiveness of flow analysis for inlining
J. Michael Ashley

August 1997 **ACM SIGPLAN Notices , Proceedings of the second ACM SIGPLAN international conference on Functional programming ICFP '97,** Volu
Issue 8
Publisher: ACM Press
Full text available: pdf(1.13 MB) Additional Information: full citation, abstract, referenc citings, index terms

An interprocedural flow analysis can justify inlining in higher-order languages. In pri more inlining can be performed as analysis accuracy improves. This paper compares f flow analyses to determine how effectively they justify inlining in practice. The paper two contributions. First, the relative merits of the flow analyses are measured with all variables held constant. The four analyses include two monovariant and two polyvari analyses that cover a wide range of the ac ...

8 Automatic inline allocation of objects
Julian Dolby
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conf on Programming language design and implementation PLDI '97,** Volum
Issue 5
Publisher: ACM Press
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, referenc citings, index terms

Object-oriented languages like Java and Smalltalk provide a uniform object model th simplifies programming by providing a consistent, abstract model of object behavior. direct implementations introduce overhead, removal of which requires aggressive implementation techniques (e.g. type inference, function specialization); in this paper introduce *object inlining*, an optimization that automatically inline allocates objects w containers (as is done by hand in C++) within a unif ...

9 Aggressive inlining
Andrew Ayers, Richard Schooler, Robert Gottlieb
May 1997 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1997 conf on Programming language design and implementation PLDI '97,** Volum
Issue 5
Publisher: ACM Press
Full text available: pdf(1.40 MB) Additional Information: full citation, abstract, referenc citings, index terms

Existing research understates the benefits that can be obtained from inlining and cloni especially when guided by profile information. Our implementation of inlining and cl yields excellent results on average and very rarely lowers performance. We believe ou results can be explained by a number of factors: inlining at the intermediate-code leve removes most technical restrictions on what can be inlined; the ability to inline across and incorporate profile information enables ...

10 Field analysis: getting useful and low-cost interprocedural information
Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conf on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, referenc citings, index terms

We present a new limited form of interprocedural analysis called field analysis that ca used by a compiler to reduce the costs of modern language features such as object-ori programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and doe require access to the entire program. Field analysis exploits the declared access restric placed on fields in a modul ...

11 Polymorphic splitting: an effective polyvariant flow analysis
Andrew K. Wright, Suresh Jagannathan
January 1998 **ACM Transactions on Programming Languages and Systems (TOPL** Volume 20 Issue 1
Publisher: ACM Press
Full text available: pdf(517.76 KB) Additional Information: full citation, abstract, referenc citings, index terms, review

This article describes a general-purpose program analysis that computes global contro and data-flow information for higher-order, call-by-value languages. The analysis em novel form of polyvariance called polymorhic splitting that uses let-expressions as sy clues to gain precision. The information derived from the analysis is used both to elim run-time checks and to inline procedure. The analysis and optimizations have been ap to a suite of Scheme progra ...

Keywords: flow analysis, inlining, polyvariance, run-time checks

12 Partitioning sequential programs for CAD using a three-step approach
Frank Vahid
July 2002 **ACM Transactions on Design Automation of Electronic Systems (TODA**
Volume 7 Issue 3
Publisher: ACM Press
Full text available: pdf(147.12 KB) Additional Information: full citation, abstract, referenc citings, index terms

Many computer-aided design problems involve solutions that require the partitioning large sequential program written in a language such as C or VHDL. Such partitioning improve design metrics such as performance, power, energy, size, input/output lines, a even CAD tool run-time and memory requirements, by partitioning among hardware modules, hardware and software processors, or even among time-slices in reconfigura computing devices. Previous partitioning approaches typically presel ...

Keywords: Partitioning, behavioral partitioning, functional partitioning, hardware/so partitioning, system level partitioning

13 Towards better inlining decisions using inlining trials
Jeffrey Dean, Craig Chambers
July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conferenc LISP and functional programming LFP '94**, Volume VII Issue 3
Publisher: ACM Press
Full text available: pdf(1.24 MB) Additional Information: full citation, abstract, referenc citings, index terms

Inlining trials are a general mechanism for making better automatic decisions about w a routine is profitable to inline. Unlike standard source-level inlining heuristics, an inl trial captures the effects of optimizations applied to the body of the inlined routine wh calculating the costs and benefits of inlining. The results of inlining trials are stored ir persistent database to be reused when making future inlining decisions at similar call : Type group analysis can d ...

14 Computing the MDMT decomposition
Linda Kaufman

December 1995 **ACM Transactions on Mathematical Software (TOMS),** Volume 21
4
Publisher: ACM Press
Full text available: pdf(829.62 KB) Additional Information: full citation, abstract, referenc citings, index terms, review

The MDMT factorization of an n×n symmetric indefinite matrix A can be used to solv linear system with A as the coefficient matrix. This factorization can be computed eff using an algorithm given in 1977 by Bunch and Kaufman. The LAPACK project has l implementing block versions of well-known algorithms for solving dense linear syste eigenvalue problems. The block version of the ...

Keywords: LAPACK, block factorization, linear systems (direct methods), symmetri indefinite

15 A graphical interval logic for specifying concurrent systems
L. K. Dillon, G. Kutty, L. E. Moser, P. M. Melliar-Smith, Y. S. Ramakrishna
April 1994 **ACM Transactions on Software Engineering and Methodology (TOSEM**
Volume 3 Issue 2
Publisher: ACM Press
Full text available: pdf(1.96 MB) Additional Information: full citation, abstract, referenc citings, index terms, review

This article describes a graphical interval logic that is the foundation of a tool set supp formal specification and verification of concurrent software systems. Experience has s that most software engineers find standard temporal logics difficult to understand and The objective of this article is to enable software engineers to specify and reason abou temporal properties of concurrent systems more easily by providing them with a logic has an intuitive graphical represe ...

Keywords: automated proof-checking, concurrent systems, formal specifications, gra interval logic, temporal logic, timing diagrams, visual languages

16 Online feedback-directed optimization of Java
Matthew Arnold, Michael Hind, Barbara G. Ryder
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, a**

applications OOPSLA '02, Volume 37 Issue 11
Publisher: ACM Press
Full text available: pdf(463.00 KB) Additional Information: full citation, abstract, referenc citings, index terms

This paper describes the implementation of an online feedback-directed optimization system. The system is fully automatic; it requires no prior (offline) profiling run. It us previously developed low-overhead instrumentation sampling framework to collect cc flow graph edge profiles. This profile information is used to drive several traditional optimizations, as well as a novel algorithm for performing feedback-directed control f graph node splitting. We empirically evaluate this syst ...

Keywords: adaptive optimization, dynamic optimization, online algorithms, virtual machines

17 A framework for interprocedural optimization in the presence of dynamic class loading
Vugranam C. Sreedhar, Michael Burke, Jong-Deok Choi
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 confe on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(576.50 KB) Additional Information: full citation, abstract, referenc citings, index terms

Dynamic class loading during program execution in the Java Programming Language impediment for generating code that is as efficient as code generated using static who program analysis and optimization. Whole-program analysis and optimization is possi languages, such as C++, that do not allow new classes and/or methods to be loaded du program execution. One solution for performing whole-program analysis and avoidin incorrect execution after a new class is loaded is to in ...

18 Compiler transformations for high-performance computing
David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4
Publisher: ACM Press
Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, referenc citings, index terms, review

In the last three decades a large number of compiler transformations for optimizing programs have been implemented. Most optimizations for uniprocessors reduce the nu of instructions executed by the program using transformations based on the analysis o scalar quantities and data-flow techniques. In contrast, optimizations for high-perform superscalar, vector, and parallel processors maximize parallelism and memory locality transformations that rely on tracking the properties o ...

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscalar processors, vectorization

19 An automatic object inlining optimization and its evaluation
Julian Dolby, Andrew Chien
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conf on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(877.17 KB) Additional Information: full citation, abstract, referenc citings, index terms

Automatic object inlining [19, 20] transforms heap data structures by fusing parent an objects together. It can improve runtime by reducing object allocation and pointer dereference costs. We report continuing work studying object inlining optimizations. particular, we present a new semantic derivation of the correctness conditions for obje inlining, and program analysis which extends our previous work. And we present an c inlining transformation, focusing ...

20 Unexpected side effects of inline substitution: a case study
Keith D. Cooper, Mary W. Hall, Linda Torczon
March 1992 **ACM Letters on Programming Languages and Systems (LOPLAS)**, V 1 Issue 1
Publisher: ACM Press
Full text available: pdf(740.92 KB) Additional Information: full citation, abstract, referenc citings, index terms, review

The structure of a program can encode implicit information that changes both the sha speed of the generated code. Interprocedural transformations like inlining often discar information; using interprocedural data-flow information as a basis for optimization c have the same effect. In the course of a study on inline substitution with commercial

FORTRAN compilers, we encountered unexpected performance problems in one of tl
programs. This paper describes the specific ...

Keywords: inline substitution, interprocedural analysis, interprocedural optimization

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free)

Search: [✓ The ACM Digital Library] The Guide

+determining +inlin* type check*

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction s

Terms used **determining** **inlin** **type** **check** Found **2,892**

Sort results by	relevance	Save results to a Binder	Try an Advanced Search
		Search Tips	Try this search in The ACM
Display results	expanded form	Open results in a new window	

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scal

1 Field analysis: getting useful and low-cost interprocedural information
Sanjay Ghemawat, Keith H. Randall, Daniel J. Scales
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conf on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(686.96 KB) Additional Information: full citation, abstract, referenc citings, index terms

We present a new limited form of interprocedural analysis called field analysis that ca used by a compiler to reduce the costs of modern language features such as object-ori programming, automatic memory management, and run-time checks required for type safety. Unlike many previous interprocedural analyses, our analysis is cheap, and doe require access to the entire program. Field analysis exploits the declared access restric placed on fields in a modul ...

2 A framework for interprocedural optimization in the presence of dynamic class loading
Vugranam C. Sreedhar, Michael Burke, Jong-Deok Choi
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conf on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(576.50 Additional Information: full citation, abstract, referenc

KB) citings, index terms

Dynamic class loading during program execution in the Java Programming Language
impediment for generating code that is as efficient as code generated using static who
program analysis and optimization. Whole-program analysis and optimization is possi
languages, such as C++, that do not allow new classes and/or methods to be loaded du
program execution. One solution for performing whole-program analysis and avoidin
incorrect execution after a new class is loaded is to in ...

3 Techniques for obtaining high performance in Java programs
Iffat H. Kazi, Howard H. Chen, Berdenia Stanley, David J. Lilja
September 2000 **ACM Computing Surveys (CSUR)**, Volume 32 Issue 3
Publisher: ACM Press
Full text available: pdf(816.13 KB) Additional Information: full citation, abstract, referenc citings, index terms

This survey describes research directions in techniques to improve the performance o
programs written in the Java programming language. The standard technique for Java
execution is interpretation, which provides for extensive portability of programs. A Ja
interpreter dynamically executes Java bytecodes, which comprise the instruction set o
Java Virtual Machine (JVM). Execution time performance of Java programs can be
improved through compilation, possibly at the expense of portabili ...

Keywords: Java, Java virtual machine, bytecode-to-source translators, direct compile
dynamic compilation, interpreters, just-in-time compilers

4 Lessons learned from the OS/400 OO project
William Berg, Marshall Cline, Mike Girou
October 1995 **Communications of the ACM**, Volume 38 Issue 10
Publisher: ACM Press
Full text available: pdf(339.92 KB) Additional Information: full citation, abstract, referenc citings, index terms

This article describes some of the lessons learned when a team of 150 developers wit
minimal prior exposure to object-oriented (OO) technology undertook a large develop
project. Team members became proficient in OO design, using C++ as an OO langua
rather than just using C++ as a better C, and developed IBM's RISC version of the AS
and System/36 operating systems from 1992 to 1994 in Rochester, Minnesota. The pr

contains 14,000 thousand classes, 90,000 thousand methods, an ...

5 Reducing virtual call overheads in a Java VM just-in-time compiler
Junpyo Lee, Byung-Sun Yang, Suhyun Kim, Kemal Ebcioğlu, Erik Altman, Seungil Le
C. Chung, Heungbok Lee, Je Hyung Lee, Soo-Mook Moon
March 2000 **ACM SIGARCH Computer Architecture News**, Volume 28 Issue 1
Publisher: ACM Press
Full text available: pdf(994.66 KB) Additional Information: full citation, abstract, index te

Java, an object-oriented language, uses *virtual methods* to support the extension and r
classes. Unfortunately, virtual method calls affect performance and thus require an ef
implementation, especially when just-in-time (JIT) compilation is done. *Inline caches*
type feedback are solutions used by compilers for dynamically-typed object-oriented
languages such as SELF [1, 2, 3], where virtual call overheads are much more critical
performance than in Java. Wi ...

Keywords: Java JIT compilation, adaptive compilation, inline cache, type feedback, v
method call

6 Interactive type analysis and extended message splitting; optimizing dynamically-typed
oriented programs
Craig Chambers, David Ungar
June 1990 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1990 conf**
on Programming language design and implementation PLDI '90, Volum
Issue 6
Publisher: ACM Press
Full text available: pdf(1.58 MB) Additional Information: full citation, abstract, referenc citings, index terms

Object-oriented languages have suffered from poor performance caused by frequent a
slow dynamically-bound procedure calls. The best way to speed up a procedure call is
compile it out, but dynamic binding of object-oriented procedure calls without static r
type information precludes inlining. Iterative type analysis and extended message spli
are new compilation techniques that extract much of the necessary type information a
make it possib ...

7 Fast subtype checking in the HotSpot JVM
Cliff Click, John Rose
November 2002 **Proceedings of the 2002 joint ACM-ISCOPE conference on Java G JGI '02**
Publisher: ACM Press
Full text available: pdf(61.60 KB) Additional Information: full citation, abstract, referenc citings, index terms

We present the fast subtype checking implemented in Sun's HotSpot JVM. Subtype cl occur when a program wishes to know if class S implements class T, where S and T a both known at compile-time. Large Java programs will make millions or even billions such checks, hence a fast check is essential. In actual benchmark runs our technique performs complete subtype checks in 3 instructions (and only 1 memory reference) essentially all the time. In rare instances it reverts to a slower array ...

Keywords: Java, checkcast, instanceof, subtype, typecase

8 Making pure object-oriented languages practical
Craig Chambers, David Ungar
November 1991 **ACM SIGPLAN Notices , Conference proceedings on Object-orien programming systems, languages, and applications OOPSLA '91,** Volume 26 Issue 11
Publisher: ACM Press
Full text available: pdf(1.86 MB) Additional Information: full citation, references, citing index terms

9 Practicing JUDO: Java under dynamic optimizations
Michał Cierniak, Guei-Yuan Lueh, James M. Stichnoth
May 2000 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conf on Programming language design and implementation PLDI '00**, Volum Issue 5
Publisher: ACM Press
Full text available: pdf(190.06 KB) Additional Information: full citation, abstract, referenc citings, index terms

A high-performance implementation of a Java Virtual Machine (JVM) consists of effi implementation of Just-In-Time (JIT) compilation, exception handling, synchronizati mechanism, and garbage collection (GC). These components are tightly coupled to ac high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Re Lab Virtual Machine (MRL VM), ...

10 Polymorphic splitting: an effective polyvariant flow analysis
Andrew K. Wright, Suresh Jagannathan
January 1998 **ACM Transactions on Programming Languages and Systems (TOPL** Volume 20 Issue 1
Publisher: ACM Press
Full text available: pdf(517.76 KB) Additional Information: full citation, abstract, referenc citings, index terms, review

This article describes a general-purpose program analysis that computes global contro and data-flow information for higher-order, call-by-value languages. The analysis em novel form of polyvariance called polymorhic splitting that uses let-expressions as sy clues to gain precision. The information derived from the analysis is used both to elim run-time checks and to inline procedure. The analysis and optimizations have been ap to a suite of Scheme progra ...

Keywords: flow analysis, inlining, polyvariance, run-time checks

11 Inline routines in VAXELN Pascal
M. Donald MacLaren
June 1984 **ACM SIGPLAN Notices , Proceedings of the 1984 SIGPLAN symposium Compiler construction SIGPLAN '84**, Volume 19 Issue 6
Publisher: ACM Press
Full text available: pdf(754.19 KB) Additional Information: full citation, abstract, referenc citings

This paper describes the implementation of inline procedures and functions inthe VA Pascal compiler. Inline expansion translates the reverse Polish text produced by the pa into an intermediate language like that used in the VAX-11 PL/I and C compilers. Th gives detailed descriptions of the front end's organization as it relates to inline routine of the symbol substitutions made during inline expansion. The paper also discusses gl optimization and the compiler's treatment ...

12 Effectiveness of cross-platform optimizations for a java just-in-time compiler
Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Go
Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue,
Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGP conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press
Full text available: pdf(405.65 KB) Additional Information: full citation, abstract, referenc citings, index terms

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, wl the basis for the latest production version of IBM Java JIT compiler that supports a di of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and V architectures. In particular, we focus on the design and evaluation of the cross-platfor optimizations that are common across different architectures. We studied the effective of each optimization by selectively disabling ...

Keywords: Java, just-in-time compiler, optimization

13 Adapting virtual machine techniques for seamless aspect support
Christoph Bockisch, Matthew Arnold, Tom Dinkelaker, Mira Mezini
October 2006 **ACM SIGPLAN Notices , Proceedings of the 21st annual ACM SIGP conference on Object-oriented programming systems, languages, and applications OOPSLA '06**, Volume 41 Issue 10
Publisher: ACM Press
Full text available: pdf(266.90 KB) Additional Information: full citation, abstract, referenc index terms

Current approaches to compiling aspect-oriented programs are inefficient. This ineffic has negative effects on the productivity of the development process and is especially prohibitive for dynamic aspect deployment. In this work, we present how well-known machine techniques can be used with only slight modifications to support fast aspect deployment while retaining runtime performance. Our implementation accelerates dyı aspect deployment by several orders of magnitude relative t ...

Keywords: aspect weaving, aspect-oriented programming, dynamic deployment, env based weaving, virtual machine support

14 TIL: a type-directed optimizing compiler for ML
D. Tarditi, G. Morrisett, P. Cheng, C. Stone, R. Harper, P. Lee
May 1996 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1996 confe on Programming language design and implementation PLDI '96,** Volum Issue 5
Publisher: ACM Press
Full text available: pdf(1.23 MB) Additional Information: full citation, references, citing index terms

15 Session 2: Extended static checking for haskell
Dana N. Xu
September 2006 **Proceedings of the 2006 ACM SIGPLAN workshop on Haskell Has '06**
Publisher: ACM Press
Full text available: pdf(233.55 KB) Additional Information: full citation, abstract, referenc index terms

Program errors are hard to detect and are costly both to programmers who spend signi efforts in debugging, and to systems that are guarded by runtime checks. Extended sta checking can reduce these costs by helping to detect bugs at compile-time, where pos Extended static checking has been applied to objectoriented languages, like Java and it has not been applied to a lazy functional language, like Haskell. In this paper, we de an extended static checking tool for Has ...

Keywords: counterexample guided unrolling, pre/postcondition, symbolic simplificat

16 Exploiting prolific types for memory management and optimizations
Yefim Shuf, Manish Gupta, Rajesh Bordawekar, Jaswinder Pal Singh
January 2002 **ACM SIGPLAN Notices , Proceedings of the 29th ACM SIGPLAN-SIGACT symposium on Principles of programming languages POPL** Volume 37 Issue 1
Publisher: ACM Press
Full text available: pdf(203.59 Additional Information: full citation, abstract, referenc

KB) citings

In this paper, we introduce the notion of *prolific* and *non-prolific* types, based on the ı of instantiated objects of those types. We demonstrate that distinguishing between the types enables a new class of techniques for memory management and data locality, an facilitates the deployment of known techniques. Specifically, we first present a new *ty based* approach to garbage collection that has similar attributes but lower cost than generational collection. Then we de ...

17 The Python compiler for CMU Common Lisp
Robert A. MacLachlan
January 1992 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1992 ACM confer on LISP and functional programming LFP '92**, Volume V Issue 1
Publisher: ACM Press
Full text available: pdf(1.06 MB) Additional Information: full citation, abstract, referenc citings, index terms

The Python compiler for CMU Common Lisp has been under development for over fi years, and now forms the core of a production quality public domain Lisp implementa Python synthesizes the good ideas from Lisp compilers and source transformation sys with mainstream optimization and retargetability techniques. Novel features include s type checking and source-level debugging of compiled code. Unusual attention has be paid to the compiler's user interface.

18 Local type inference
Benjamin C. Pierce, David N. Turner
January 2000 **ACM Transactions on Programming Languages and Systems (TOPL** Volume 22 Issue 1
Publisher: ACM Press
Full text available: pdf(359.69 KB) Additional Information: full citation, abstract, referenc citings, index terms, review

We study two partial type inference methods for a language combining subtyping and impredicative polymorphism. Both methods are local in the sense that missing annota are recovered using only information from adjacent nodes in the syntax tree, without l distance constraints such as unification variables. One method infers type arguments i polymorphic applications using a local constraint solver. The other infers annotations bound variables in function abstractio ...

Keywords: polymorphism, subtyping, type inference

19 Compiling functional languages with flow analysis
Suresh Jagannathan, Andrew Wright
June 1996 **ACM Computing Surveys (CSUR)**, Volume 28 Issue 2
Publisher: ACM Press
Full text available: pdf(162.07 KB) Additional Information: full citation, references, index

20 Compiling nested data-parallel programs for shared-memory multiprocessors
Siddhartha Chatterjee
July 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)** Volume 15 Issue 3
Publisher: ACM Press
Full text available: pdf(4.17 MB) Additional Information: full citation, references, citing index terms, review

Keywords: compilers, data parallelism, shared-memory multiprocessors

Results 1 - 20 of 200		Result page: **1** 2 3 4 5 6 7 8 9 10 next

Home | Login | Logout | Access

IEEE Xplore® RELEASE 2.3

Welcome United States Patent and Trademark Office

Search Results

BROWSE SEARCH IEEE XPLORE GUIDE SI

Results for "(((inlin* type check*)<in>metadata)) <and> (pyr >= 1980 <and> pyr <= 2003)"
Your search matched **0** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History
New Search

Modify Search

(((inlin* type check*)<in>metadata)) <and> (pyr >= 1980 <and> p

☐ Check to search only within this results set

Display Format: [✓ Citation] Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IET JNL	IET Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IET CNF	IET Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help p assistance revising your search.

Help

Indexed by Inspec

Home | Login | Logout | Access

IEEE Xplore RELEASE 2.3

Welcome United States Patent and Trademark Office

Search Results

BROWSE SEARCH IEEE XPLORE GUIDE **SI**

Results for **"(((where to insert type checking code)<in>metadata)) <and> (pyr >= 1980 <and> p...**
Your search matched **0** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(((where to insert type checking code)<in>metadata)) <and> (pyr :

☐ Check to search only within this results set

Display Format: [✓ Citation] ○ Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IET JNL	IET Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IET CNF	IET Conference Proceeding
IEEE STD	IEEE Standard

No results were found.

Please edit your search criteria and try again. Refer to the Help p assistance revising your search.

Help

Indexed by Inspec

© Co